HNRS 190

On the Halting Problem

James Mendel

December 2019

1 Introduction

In 1900, German mathematician David Hilbert published 23 mathematical problems, systematically referred to today as Hilbert's problems. This paper will focus on the effects of one of those problems, and its relevance in computer theory and modern computing systems. The halting problem asks simple question: is it possible to know, with complete certainty, whether or not a computation is possible before attempting the computation itself [2]? Allan Turing was notably one of the first to conclude that this is an impossible feat to accomplish.

2 Literature Survey

At the time of his initial proposal, David Hilbert had no interest in solving the question as it is commonly framed in computing. The concept of computers as they are known today had yet to come to fruition. It was his tenth problem, titled, "DETERMINATION OF THE SOLVABILITY OF A BIOPHANTINE EQUATION" that asked if it was possible,

> To devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers. [2, 447]

Later, in 1928, Hilbert published another paper alongside Wilhelm Ackermann that had an outline for a problem with very similar roots and consequences. Called the *Entschei-*

dungsproblem, or *Decision Problem*, it asks specifically if an algorithm can be devised to determine weather or not a statement is universally valid.

The advantages of being able to accomplish this would be seen not only in the computing world, but they would also cascade down to the original, mathematical logic-based problem. The mathematics world would benefit greatly in being able to know the possibility of completing a computation before endeavoring a journey of inevitable failure. Kurt Gödel was one of the first mathematicians to spark interest in Hilbert's *Entscheidungsproblem* in his proof that no algorithm can sort out all true or untrue expressions within the arithmetic system. In other words, there are expressions that exist that can not be proven within the arithmetic system [1]. This proof sparked a profound interest in the problem in the rest of the mathematics community.

Alonzo Church was the first to disprove the decision problem, devising λ-calculus in the process. Turing picked up the problem soon after and proved the *Entscheidungsproblem* with his the newly birthed concept of a Turing machine. It was here that the problem gained its significance in modern computing, colloquially referred to as the halting problem. The halting problem, also known as the *Spinning Beach-ball* or *Spinning Pinwheel* problem can be rephrased from Hilbert's original problem once more: can it be determined whether or not a computation process will succeed or fail (halt) without computing the process itself.

Figure 1: Spinning Pinwheel from Mac OS

In Turing's paper, ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM, he notably outlines the Turing machine. This description was before the first digital computer. It can fundamentally describe any piece of software and is the basis of today's computing.

Turing's definition of a "computing machine" is that of the following: a strip of paper (tape) is fed into an automatic machine. The paper is divided into squares and the contents of the square is read by the machine. A square can either be blank, or contain a symbol (instructions or data). The symbol should be read by the machine to dictate one of the following actions:

1. Move the read/write head left or right
2. Read the current square
3. Write/overwrite a square

The action that the machine takes is determined by a table of rules, where all possible inputs, and their actions are defined [4]. He then shows the relation between his computing machine and the decision problem. He states that using an example he outlines prior, it "can be used to show that the Hilbert Entscheidungsproblem can have no solution" [4, 259].

2.1 Proof

The general proof, modernized beyond the bounds of a Turing machine, can be described with the following pseudocode and explanation:

```
Z = Program(String x)
if (halt(x, x)){
    loop(); //x halts
}
else{
    return false; //x does not halt
}
```

Where `halt(P, i)` is a function that will return true if the program, P, halts on a given input, i. Turing proved that the halting problem is not solvable through contradiction. In the code above, running it with input Z ($x = Z$) results in the following contradiction:

1. **Case 1**: Program Z halts when given input Z.
   (a) halt() will return true on input (Z, Z).
   (b) Program Z now loops forever, therefore, it has not halted. Contradiction.

(c) Program Z loops forever with input Z.

(d) halt() will now return false on input (Z, Z). This is a contradiction to the original statement. [3]

This logical contradiction proves the unreliability of they concept proposed by Hilbert.

2.2 Further Consideration

Eugene Eberbach mentions a method of solving the halting problem when taken outside of the realm of Turing Machine.

He explains the difficulty in convincing the computer science community that this problem can be viewed outside the context of a Turing Machine. He points out the shift occurring between the definition of a Turing Machine, as explained by Turing himself, and modern computing. He remarks:

> Do real computer and TMs define the same set of computable things? According to "classical" computer science textbooks the answer is "yes". In this paper, we provide some arguments that the answer should be rather "no", and TM is only an important approximation of what a large class of discrete computers can do. [1, 3]

He concludes that by using models more expressive than Turing Machines, it should be possible to solve the halting problem. Accomplishing this would be a great feat in the mathematics and computing world, however it has yet to occur. See Eberbach's article *Is Entscheidungsproblem Solvable?* [1] for further reading.

References

[1] E. Eberbach, “Is entscheidungsproblem solvable? beyond undecidability of turing machines and its consequence for computer science and mathematics,” *Computational Mathematics, Modelling and Algorithms*, pp. 1–32, 2003.

An explanation of the *Entscheidungsproblem* in context of modern computing.

[2] D. Hilbert, “Mathematical problems,” *Bull. Amer. Math. Soc.*, vol. 8, no. 10, pp. 437–479, 1902.

The original text of Hilbert’s 23 problems.

[3] K. Pruhs, “Halting problem – simple proof,” Nov 1997. [Online]. Available: https://www.comp.nus.edu.sg/~cs5234/FAQ/halt.html

A simplified proof of the halting problem.

[4] A. M. Turing, “On computable numbers, with an application to the entscheidungsproblem,” *Proceedings of the London mathematical society*, vol. 2, no. 1, pp. 230–265, 1937.

Turing’s paper illustrating the concept of a Turing machine and outlining his proof of the halting problem’s uncertainty.